Biologically Inspired Computing

# EECS 6180

# Homework 6

# Neural Evolution

**Edris Amin**

April 5, 2012

## Introduction

A 4 layer NN and 5 layer NN were evolved to compare to the *Iris_bp* algorithm. The programs for all algorithms are provided in the appendix. Figure 1 is a sample NN with only 3 hidden neurons. Figure 2 and 3 represent a sample chromosome corresponding to the 4 and 5 hidden neuron evolutionary NN's designed for this project.

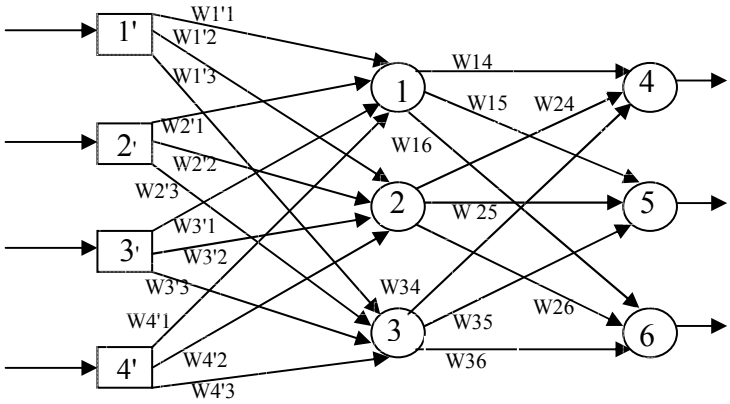


Fig. 1: Sample Neural network Architecture for Iris Data Set

| W1'1 | W1'2 | W1'3 | W1'4 | W2'1 | W2'2 | W2'3 | W2'4 | W3'1 | W3'2 | W3'3 | W3'4 | W4'1 | W4'2 | W4'3 | W4'4 | W15 | W16 | W17 | W25 | W26 | W27 | W35 | W36 | W37 | W45 | W46 | W47 | T1 | T2 | T3 | T4 | T5 | T6 | T7 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

Fig. 2: Chromosome 4 hidden neuron NN

| W1'1 | W1'2 | W1'3 | W1'4 | W1'5 | W2'1 | W2'2 | W2'3 | W2'4 | W2'5 | W3'1 | W3'2 | W3'3 | W3'4 | W3'5 | W4'1 | W4'2 | W4'3 | W4'4 | W4'5 | W16 | W17 | W18 | W26 | W27 | W28 | W36 | W37 | W38 | W46 | W47 | W48 | W56 | W57 | W58 | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

Fig. 3: Chromosome 4 hidden neuron NN

## Technique

1. Training

In each case a set of 100 weight matrices were generated for Evolutionary training. This was done with:

```
W=rand(100, 35); for 4 hidden Neurons
W=rand(100, 43); for 5 hidden Neurons
```

The training and testing data were randomly selected from *Iris_data.* The training algorithm was a for loop which used a feed forward technique with a sigmoid function. The error for each input and subsequent set of weights (individual chromosome) was used to select the top 40 weight chromosomes.

2. Crossover

The top 40 weight chromosomes were crossed such that only weights corresponding to hidden1, hidden2…, hidden5, out1, out2, and out3 crossed with one another and genes from hidden1 did not cross with any other neuron's genes.

3. Mutation

The top 60 weight chromosomes were used for mutation and each individual gene had a probability of mutation = 0.1%

After each training generation if the mse of the training output was less than 0.07 then training would stop.

4. Testing

After either the training had run out of training samples or the training criteria was met then the testing was done in the same fashion only using the best weight chromosome determined from training.

**Results**

| | | | Testing Error | | |
|---|---|---|---|---|---|
| Method | Training time (s) | Epochs generations | %Error Iris-setosa | %Error Iris-versicolor | %Error Iris-verginica |
| Iris_bp | 6.1347 | 1000 | 0% | 10% | 0% |
| Evo NN 4hidden | 6.65 | 100 | 40% | 100% | 100% |
| Evo NN 5hidden | 6.45 | 99 | 6.25% | 0% | 100% |

**Conclusion**

The results show that there is more time needed to develop a good working Evolutionary Neural Network, but it is possible to obtain a better performance with respect to time.

```
function [iris_data] = Iris_data()

% ======================================
% Data set for Iris plant classification
% ======================================

% ============================================================================
% Reference: Negnevitsky, M., "Artificial Intelligence: A Guide to Intelligent
%            Systems", 2nd edn. Addison Wesley, Harlow, England, 2005.
%            Sec. 9.4 Will a neural network work for my problem?
% ============================================================================

% Iris plant data set

  iris_data = [5.1 3.5 1.4 0.2   4.9 3.0 1.4 0.2   % Iris-setosa
               4.7 3.2 1.3 0.2   4.6 3.1 1.5 0.2   % Iris-setosa
               5.0 3.6 1.4 0.2   5.4 3.9 1.7 0.4   % Iris-setosa
               4.6 3.4 1.4 0.3   5.0 3.4 1.5 0.2   % Iris-setosa
               4.4 2.9 1.4 0.2   4.9 3.1 1.5 0.1   % Iris-setosa
               5.4 3.7 1.5 0.2   4.8 3.4 1.6 0.2   % Iris-setosa
               4.8 3.0 1.4 0.1   4.3 3.0 1.1 0.1   % Iris-setosa
               5.8 4.0 1.2 0.2   5.7 4.4 1.5 0.4   % Iris-setosa
               5.4 3.9 1.3 0.4   5.1 3.5 1.4 0.3   % Iris-setosa
               5.7 3.8 1.7 0.3   5.1 3.8 1.5 0.3   % Iris-setosa
               5.4 3.4 1.7 0.2   5.1 3.7 1.5 0.4   % Iris-setosa
               4.6 3.6 1.0 0.2   5.1 3.3 1.7 0.5   % Iris-setosa
               4.8 3.4 1.9 0.2   5.0 3.0 1.6 0.2   % Iris-setosa
               5.0 3.4 1.6 0.4   5.2 3.5 1.5 0.2   % Iris-setosa
               5.2 3.4 1.4 0.2   4.7 3.2 1.6 0.2   % Iris-setosa
               4.8 3.1 1.6 0.2   5.4 3.4 1.5 0.4   % Iris-setosa
               5.2 4.1 1.5 0.1   5.5 4.2 1.4 0.2   % Iris-setosa
               4.9 3.1 1.5 0.1   5.0 3.2 1.2 0.2   % Iris-setosa
               5.5 3.5 1.3 0.2   4.9 3.1 1.5 0.1   % Iris-setosa
               4.4 3.0 1.3 0.2   5.1 3.4 1.5 0.2   % Iris-setosa
               5.0 3.5 1.3 0.3   4.5 2.3 1.3 0.3   % Iris-setosa
               4.4 3.2 1.3 0.2   5.0 3.5 1.6 0.6   % Iris-setosa
               5.1 3.8 1.9 0.4   4.8 3.0 1.4 0.3   % Iris-setosa
               5.1 3.8 1.6 0.2   4.6 3.2 1.4 0.2   % Iris-setosa
               5.3 3.7 1.5 0.2   5.0 3.3 1.4 0.2   % Iris-setosa
               7.0 3.2 4.7 1.4   6.4 3.2 4.5 1.5   % Iris-versicolor
               6.9 3.1 4.9 1.5   5.5 2.3 4.0 1.3   % Iris-versicolor
               6.5 2.8 4.6 1.5   5.7 2.8 4.5 1.3   % Iris-versicolor
               6.3 3.3 4.7 1.6   4.9 2.4 3.3 1.0   % Iris-versicolor
               6.6 2.9 4.6 1.3   5.2 2.7 3.9 1.4   % Iris-versicolor
               5.0 2.0 3.5 1.0   5.9 3.0 4.2 1.5   % Iris-versicolor
               6.0 2.2 4.0 1.0   6.1 2.9 4.7 1.4   % Iris-versicolor
               5.6 2.9 3.6 1.3   6.7 3.1 4.4 1.4   % Iris-versicolor
               5.6 3.0 4.5 1.5   5.8 2.7 4.1 1.0   % Iris-versicolor
               6.2 2.2 4.5 1.5   5.6 2.5 3.9 1.1   % Iris-versicolor
               5.9 3.2 4.8 1.8   6.1 2.8 4.0 1.3   % Iris-versicolor
               6.3 2.5 4.9 1.5   6.1 2.8 4.7 1.2   % Iris-versicolor
```



```
               6.4 2.9 4.3 1.3   6.6 3.0 4.4 1.4   % Iris-versicolor
               6.8 2.8 4.8 1.4   6.7 3.0 5.0 1.7   % Iris-versicolor
               6.0 2.9 4.5 1.5   5.7 2.6 3.5 1.0   % Iris-versicolor
               5.5 2.4 3.8 1.1   5.5 2.4 3.7 1.0   % Iris-versicolor
               5.8 2.7 3.9 1.2   6.0 2.7 5.1 1.6   % Iris-versicolor
               5.4 3.0 4.5 1.5   6.0 3.4 4.5 1.6   % Iris-versicolor
               6.7 3.1 4.7 1.5   6.3 2.3 4.4 1.3   % Iris-versicolor
               5.6 3.0 4.1 1.3   5.5 2.5 4.0 1.3   % Iris-versicolor
               5.5 2.6 4.4 1.2   6.1 3.0 4.6 1.4   % Iris-versicolor
               5.8 2.6 4.0 1.2   5.0 2.3 3.3 1.0   % Iris-versicolor
               5.6 2.7 4.2 1.3   5.7 3.0 4.2 1.2   % Iris-versicolor
               5.7 2.9 4.2 1.3   6.2 2.9 4.3 1.3   % Iris-versicolor
               5.1 2.5 3.0 1.1   5.7 2.8 4.1 1.3   % Iris-versicolor
               6.3 3.3 6.0 2.5   5.8 2.7 5.1 1.9   % Iris-verginica
               7.1 3.0 5.9 2.1   6.3 2.9 5.6 1.8   % Iris-verginica
               6.5 3.0 5.8 2.2   7.6 3.0 6.6 2.1   % Iris-verginica
               4.9 2.5 4.5 1.7   7.3 2.9 6.3 1.8   % Iris-verginica
               6.7 2.5 5.8 1.8   7.2 3.6 6.1 2.5   % Iris-verginica
               6.5 3.2 5.1 2.0   6.4 2.7 5.3 1.9   % Iris-verginica
               6.8 3.0 5.5 2.1   5.7 2.5 5.0 2.0   % Iris-verginica
               5.8 2.8 5.1 2.4   6.4 3.2 5.3 2.3   % Iris-verginica
               6.5 3.0 5.5 1.8   7.7 3.8 6.7 2.2   % Iris-verginica
               7.7 2.6 6.9 2.3   6.0 2.2 5.0 1.5   % Iris-verginica
               6.9 3.2 5.7 2.3   5.6 2.8 4.9 2.0   % Iris-verginica
               7.7 2.8 6.7 2.0   6.3 2.7 4.9 1.8   % Iris-verginica
               6.7 3.3 5.7 2.1   7.2 3.2 6.0 1.8   % Iris-verginica
               6.2 2.8 4.8 1.8   6.1 3.0 4.9 1.8   % Iris-verginica
               6.4 2.8 5.6 2.1   7.2 3.0 5.8 1.6   % Iris-verginica
               7.4 2.8 6.1 1.9   7.9 3.8 6.4 2.0   % Iris-verginica
               6.4 2.8 5.6 2.2   6.3 2.8 5.1 1.5   % Iris-verginica
               6.1 2.6 5.6 1.4   7.7 3.0 6.1 2.3   % Iris-verginica
               6.3 3.4 5.6 2.4   6.4 3.1 5.5 1.8   % Iris-verginica
               6.0 3.0 4.8 1.8   6.9 3.1 5.4 2.1   % Iris-verginica
               6.7 3.1 5.6 2.4   6.9 3.1 5.1 2.3   % Iris-verginica
               5.8 2.7 5.1 1.9   6.8 3.2 5.9 2.3   % Iris-verginica
               6.7 3.3 5.7 2.5   6.7 3.0 5.2 2.3   % Iris-verginica
               6.3 2.5 5.0 1.9   6.5 3.0 5.2 2.0   % Iris-verginica
               6.2 3.4 5.4 2.3   5.9 3.0 5.1 1.8]; % Iris-verginica
```



```
% ==================
% Filename: Iris_bp.m
% ==================

echo off

disp(' ====================================================' )
disp(' Iris plant classification: back-propagation algorithm' )
disp(' ====================================================' )

disp(' ============================================================================' )
disp(' Reference: Negnevitsky, M., "Artificial Intelligence: A Guide to Intelligent' )
disp('            Systems", 2nd edn. Addison Wesley, Harlow, England, 2005.        ' )
disp('            Sec. 9.4 Will a neural network work for my problem?              ' )
disp(' ============================================================================' )

disp('↙
============================================================================' )
disp(' Problem: The Iris plant data set contains 3 classes, and each class is represented ↙
')
disp('          by 50 plants. A plant is characterised by its sepal length, sepal width, ↙
')
disp('          petal length and petal width. A three-layer back-propagation network is ↙
')
disp('          required to classify Iris plants. ↙
')
disp('↙
============================================================================' )

[iris_data] = Iris_data;
iris_data = (iris_data(:,[1:4]))';

% Massaged values for the Iris plant data set

for n=1:4;
    iris_inputs(n,:)=(iris_data(n,:)-min(iris_data(n,:)))/ ...
        (max(iris_data(n,:)-min(iris_data(n,:))));
end

iris_target1 = [1 0 0]'; setosa=find(iris_target1);
iris_target2 = [0 1 0]'; versicolor=find(iris_target2);
iris_target3 = [0 0 1]'; verginica=find(iris_target3);

for n=1:(50-1)
   iris_target1=[iris_target1 iris_target1(:,1)];
   iris_target2=[iris_target2 iris_target2(:,1)];
   iris_target3=[iris_target3 iris_target3(:,1)];
end

iris_targets = [iris_target1 iris_target2 iris_target3];
```



```
disp('Hit any key to randomly select input vectors to be used in training.' )
disp(' ')
pause

p=[]; t=[]; test_p=[]; test_t=[];

for n=1:150
   if rand(1)>1/3
      p=[p iris_inputs(:,n)];
      t=[t iris_targets(:,n)];
   else
      test_p=[test_p iris_inputs(:,n)];
      test_t=[test_t iris_targets(:,n)];
   end
end

[m n]=size(test_p);

disp(' ')
fprintf(1,' The training data set contains %.0f elements.\n',(150-n));
fprintf(1,' The test data set contains %.0f elements.\n',n);
disp(' ')

echo on

% Hit any key to define the network architecture.
pause

s1=5; % Five neurons in the hidden layer
s2=3; % Three neuron in the output layer

% Hit any key to create the network, initialise its weights and biases,
% and set up training parameters.
pause

rand('seed',1243);

net = newff([4.3 7.9; 2.0 4.4; 1.0 6.9; 0.1 2.5],[s1 s2],{ 'logsig'↙
'purelin'},'traingdx');

net.trainParam.show=20;       % Number of epochs between showing the progress
net.trainParam.epochs=1000;   % Maximum number of epochs
net.trainParam.goal=0.001;    % Performance goal
net.trainParam.lr=0.1;        % Learning rate
net.trainParam.lr_inc=1.05;   % Learning rate increase multiplier
net.trainParam.lr_dec=0.7;    % Learning rate decrease multiplier
net.trainParam.mc=0.9;        % Momentum constant

% Hit any key to train the back-propagation network.
pause
```

```
net=train(net,p,t);

echo off

disp(' ')
fprintf(1,' Iris-setosa is represented by output:      %.0f \n',setosa);
fprintf(1,' Iris-versicolor is represented by output:  %.0f \n',versicolor);
fprintf(1,' Iris-verginica is represented by output:   %.0f \n',verginica);

disp(' ')
disp(' Hit any key to test the network using the test data set.')
disp(' ')
pause

n_setosa=0; n_versicolor=0; n_verginica=0;
error_setosa=0; error_versicolor=0; error_verginica=0; error=0;

fprintf(' Sepal length  Sepal width  Petal length  Petal width  Desired output   Actual output  Error\n');

for i=1:n
   fprintf('     %.1f           %.1f          %.1f           %.1f',test_p(1,i),test_p(2,i),test_p(3,i),test_p(4,i));
   a=compet(sim(net,test_p(:,i))); a=find(a);
   b=compet(test_t(:,i)); b=find(b);
   if b==1
      n_setosa=n_setosa+1;
      fprintf('      Iris-setosa            ');
      if abs(a-b)>0
         error_setosa=error_setosa+1;
         fprintf('%.0f        Yes\n',a);
      else
         fprintf('%.0f        No\n',a);
      end
   elseif b==2
      n_versicolor=n_versicolor+1;
      fprintf('      Iris-versicolor        ');
      if abs(a-b)>0
         error_versicolor=error_versicolor+1;
         fprintf('%.0f        Yes\n',a);
      else
         fprintf('%.0f        No\n',a);
      end
   else
      n_verginica=n_verginica+1;
      fprintf('      Iris-verginica         ');
      if abs(a-b)>0
         error_verginica=error_verginica+1;
         fprintf('%.0f        Yes\n',a);
      else
         fprintf('%.0f        No\n',a);
```



```
      end
   end
end

error=(error_setosa+error_versicolor+error_verginica)/n*100;

error_setosa=error_setosa/n_setosa*100;
error_versicolor=error_versicolor/n_versicolor*100;
error_verginica=error_verginica/n_verginica*100;

fprintf(1,' \n')
fprintf(1,' Iris-setosa recognition error:      %.2f \n',error_setosa);
fprintf(1,' Iris-versicolor recognition error:  %.2f \n',error_versicolor);
fprintf(1,' Iris-verginica recognition error:   %.2f \n',error_verginica);
fprintf(1,' \n')
fprintf(1,' Total Iris plant recognition error: %.2f \n',error);
fprintf(1,' \n')

disp('end of Iris_bp.m')
```



```
%Edris Evolutionary NN
%inputs = 4
%weights = 16

%hidden = 4
%bias = 4
%weights = 12

%out = 3
%bias = 3

%total weights = 35
% function [msE] = myNN5(W)

% W=rand(100, 27);

echo off;
rng('default');
W=rand(100, 35);
W1 = W;

%input training data
[iris_data] = Iris_data;
iris_data = (iris_data(:,[1:4]))';

for n=1:4; %scaling
    iris_inputs(n,:)=(iris_data(n,:)-min(iris_data(n,:)))/ ...
        (max(iris_data(n,:)-min(iris_data(n,:))));
end

%target training data
iris_target1 = [1 0 0]'; setosa=find(iris_target1);
iris_target2 = [0 1 0]'; versicolor=find(iris_target2);
iris_target3 = [0 0 1]'; verginica=find(iris_target3);

for n=1:(50-1)
   iris_target1=[iris_target1 iris_target1(:,1)];
   iris_target2=[iris_target2 iris_target2(:,1)];
   iris_target3=[iris_target3 iris_target3(:,1)];
end

iris_targets = [iris_target1 iris_target2 iris_target3];

p=[]; t=[]; test_p=[]; test_t=[]; setosaind=[]; versicolorind=[]; verginicaind=[]; amsE=[];

testc = 0;
for n=1:150
   if rand(1)>1/3   %trainging data
      p=[p iris_inputs(:,n)];
      t=[t iris_targets(:,n)];
```



```
   else            %testing data
      testc = testc+1;  %increment test index
      test_p=[test_p iris_inputs(:,n)];
      test_t=[test_t iris_targets(:,n)];
      if n<51
          setosaind=[setosaind n];
          endsetosa = testc; %index of test where setosaind ends
      elseif n<101
          versicolorind=[versicolorind n];
          endvcolor = testc; %index of test where versicolorind ends
      else
          verginicaind=[verginicaind n];
          endverg = testc;   %index of test where verginicaind ends
      end
   end
end
[a nsetosa] = size(setosaind)
[a nvcolor] = size(versicolorind)
[a nverg] = size(verginicaind)
% %P is input data for NN
% P = iris_data;
%
% %T is training output for NN
% T = vertcat(iris_target1', iris_target2', iris_target3')';

[m n]=size(test_p);

trainingsize = 150 - n;
testingsize = n;

disp(' ')
fprintf(1,' The training data set contains %.0f elements.\n',(150-n));
fprintf(1,' The test data set contains %.0f elements.\n',n);
disp(' ')

chromo = 1;
train = 1;
tic;
for train = 1:trainingsize;
    for chromo = 1:100;
        %weights (to, from)
        %   [row 1 = from first layer
        %    column 1 = to first layer]
        % to hidden[1,4] from input[1,3]
        %   [          W(1)                    W(2)                    W(3)
W(4);          W(29)    %hidden1
        %              W1'1                    W2'1                    W3'1
W4'1;        bias
        h1=p(1,train)*W(chromo,1)+ p(2,train)*W(chromo,2)+ p(2,train)*W(chromo,3)+ p(4,train)*W(chromo,4)+W(chromo,29);
        z1= 1/(1+exp(-h1));
```

```
        %                   W(5)                W(6)                    W(7) ↙
W(8);           W(30)   %hidden2
        %                   W1'2                W2'2                    W3'2 ↙
W4'2;           bias
        h2=p(1,train)*W(chromo,5)+ p(2,train)*W(chromo,6)+ p(2,train)*W(chromo,7)+ p(4, ↙
train)*W(chromo,8)+W(chromo,30);
        z2= 1/(1+exp(-h2));

        %    W(9) W(10) W(11) W(12); W(31)   %hidden3
        %    W1'3 W2'3 W3'3 W4'3;   bias
        h3=p(1,train)*W(chromo,9)+ p(2,train)*W(chromo,10)+ p(2,train)*W(chromo,11)+ p(4, ↙
train)*W(chromo,12)+W(chromo,31);
        z3= 1/(1+exp(-h3));

        %    W(13) W(14) W(15) W(16);  W(32)    %hidden4
        %    W1'4 W2'4 W3'4 W4'4;   bias
        h4=p(1,train)*W(chromo,13)+ p(2,train)*W(chromo,14)+ p(2,train)*W(chromo,15)+ p ↙
(4,train)*W(chromo,16)+W(chromo,32);
        z4= 1/(1+exp(-h4));

        %to output (layer 3) from hidden (layer 2)
        %              [W15                 W25             W35              W45 ↙
T5; out1
        %               W(17)               W(18)           W(19)           W(20) ↙
W(33);
        out1 = z1*W(chromo, 17) + z2*W(chromo, 18)+ z3*W(chromo, 19)+ z4*W(chromo, 20)+ W ↙
(chromo, 33);

        %                   W16              W26              W36             W46 ↙
T6;
        %                  W(21)            W(22)            W(23)          W(24) ↙
W(34); out2
        out2 = z1*W(chromo, 21) + z2*W(chromo, 22)+ z3*W(chromo, 23)+ z4*W(chromo, 24)+ W ↙
(chromo, 34);

        %                  W17              W27              W37              W47 ↙
T7];
        %                 W(25)            W(26)            W(27)            W(28) ↙
W(35)];
        out3 = z1*W(chromo, 25) + z2*W(chromo, 26)+ z3*W(chromo, 27)+ z4*W(chromo, 28)+ W ↙
(chromo, 35);

        %get simulated outputs Y from defined NN net with defined weights W,
        %and input P
        Y = [out1; out2; out3];

        %Error = expected - simulated = T - Y
        E(chromo,:) = t(:,train) - Y;
        msE(chromo,:) = mse(E(chromo,:)); %mean squared error
    end %end population
```



```
    Wnext = zeros(100, 35); %next population
    Esearch = msE;    %used for selection to find smallest error

    amsE = [amsE, train];
    if train > 1
        if amsE(train) < 0.07
            break %stop training
        end
    end

    %selection order of E index (least E) chromosomes-----------------
    %this array order least mse to worst mse indcies
    %which coorespond to W indecies
    for i = 1:100;
        [val, ind] = min(Esearch);
        sel(i,1) = ind;
        Esearch(ind, 1) = 100;
    end
%     sel; %array of indecies of E order min - max

    %crossover-----------------------------
    pcross = 0.2;
    Xind = randperm(40); %select order of top 40 indcies to be crossed
    Xgene = rand(7,1); %probablility or crossover per gene one gene per hidden per output
    for n = 1:20;
        for gene = 1:7;
            if gene == 1;    %hidden 1 1:4; 29
                if Xgene(gene, 1) < pcross; %cross
                    Wnext( (Xind(2*n-1)), 1:4)  = W( sel(Xind(2*n)), 1:4);
                    Wnext( (Xind(2*n-1)), 29)   = W( sel(Xind(2*n)), 29);
                    Wnext( (Xind(2*n)), 1:4)    = W( sel(Xind(2*n-1)), 1:4);
                    Wnext( (Xind(2*n)), 29)     = W( sel(Xind(2*n-1)), 29);
                else %copy
                    Wnext( (Xind(2*n-1)), 1:4) = W( sel(Xind(2*n-1)), 1:4);
                    Wnext( (Xind(2*n-1)), 29) = W( sel(Xind(2*n-1)), 29);
                    Wnext( (Xind(2*n)), 1:4) = W( sel(Xind(2*n)), 1:4);
                    Wnext( (Xind(2*n)), 29) = W( sel(Xind(2*n)), 29);

                end

            elseif gene == 2;    %hidden 2 5:8; 30
                if Xgene(gene, 1) < pcross; %cross
                    Wnext( (Xind(2*n-1)), 5:8) = W( sel(Xind(2*n)), 5:8);
                    Wnext( (Xind(2*n-1)), 30) = W( sel(Xind(2*n)), 30);
                    Wnext( (Xind(2*n)), 5:8) = W( sel(Xind(2*n-1)), 5:8);
                    Wnext( (Xind(2*n)), 30) = W( sel(Xind(2*n-1)), 30);
                else %copy
                    Wnext( (Xind(2*n-1)), 5:8) = W( sel(Xind(2*n-1)), 5:8);
                    Wnext( (Xind(2*n-1)), 30) = W( sel(Xind(2*n-1)), 30);
                    Wnext( (Xind(2*n)), 5:8) = W( sel(Xind(2*n)), 5:8);
```



```
                    Wnext( (Xind(2*n)), 30) = W( sel(Xind(2*n)), 30);
                end

            elseif gene == 3;    %hidden 3 9:12; 31;
                if Xgene(gene, 1) < pcross; %cross
                    Wnext( (Xind(2*n-1)), 9:12) = W( sel(Xind(2*n)), 9:12);
                    Wnext( (Xind(2*n-1)), 31) = W( sel(Xind(2*n)), 31);
                    Wnext( (Xind(2*n)), 9:12) = W( sel(Xind(2*n-1)), 9:12);
                    Wnext( (Xind(2*n)), 31) = W( sel(Xind(2*n-1)), 31);
                else %copy
                    Wnext( (Xind(2*n-1)), 9:12) = W( sel(Xind(2*n-1)), 9:12);
                    Wnext( (Xind(2*n-1)), 31) = W( sel(Xind(2*n-1)), 31);
                    Wnext( (Xind(2*n)), 9:12) = W( sel(Xind(2*n)), 9:12);
                    Wnext( (Xind(2*n)), 31) = W( sel(Xind(2*n)), 31);
                end

            elseif gene == 4;    %hidden 4 13:16; 32
               if Xgene(gene, 1) < pcross; %cross
                    Wnext( (Xind(2*n-1)), 13:16) = W( sel(Xind(2*n)), 13:16);
                    Wnext( (Xind(2*n-1)), 32) = W( sel(Xind(2*n)), 32);
                    Wnext( (Xind(2*n)), 13:16) = W( sel(Xind(2*n-1)), 13:16);
                    Wnext( (Xind(2*n)), 32) = W( sel(Xind(2*n-1)), 32);
                else %copy
                    Wnext( (Xind(2*n-1)), 13:16) = W( sel(Xind(2*n-1)), 13:16);
                    Wnext( (Xind(2*n-1)), 32) = W( sel(Xind(2*n-1)), 32);
                    Wnext( (Xind(2*n)), 13:16) = W( sel(Xind(2*n)), 13:16);
                    Wnext( (Xind(2*n)), 32) = W( sel(Xind(2*n)), 32);
                end

            elseif gene == 5;    %out 1 17:20; 33
               if Xgene(gene, 1) < pcross; %cross
                    Wnext( (Xind(2*n-1)), 17:20) = W( sel(Xind(2*n)), 17:20);
                    Wnext( (Xind(2*n-1)), 33) = W( sel(Xind(2*n)), 33);
                    Wnext( (Xind(2*n)), 17:20) = W( sel(Xind(2*n-1)), 17:20);
                    Wnext( (Xind(2*n)), 33) = W( sel(Xind(2*n-1)), 33);
                else %copy
                    Wnext( (Xind(2*n-1)), 17:20) = W( sel(Xind(2*n-1)), 17:20);
                    Wnext( (Xind(2*n-1)), 33) = W( sel(Xind(2*n-1)), 33);
                    Wnext( (Xind(2*n)), 17:20) = W( sel(Xind(2*n)), 17:20);
                    Wnext( (Xind(2*n)), 33) = W( sel(Xind(2*n)), 33);
               end

            elseif gene == 6;    %out2 21:24; 34
                if Xgene(gene, 1) < pcross; %cross
                    Wnext( (Xind(2*n-1)), 21:24) = W( sel(Xind(2*n)), 21:24);
                    Wnext( (Xind(2*n-1)), 34) = W( sel(Xind(2*n)), 34);
                    Wnext( (Xind(2*n)), 21:24) = W( sel(Xind(2*n-1)), 21:24);
                    Wnext( (Xind(2*n)), 34) = W( sel(Xind(2*n-1)), 34);
                else %copy
                    Wnext( (Xind(2*n-1)), 21:24) = W( sel(Xind(2*n-1)), 21:24);
                    Wnext( (Xind(2*n-1)), 34) = W( sel(Xind(2*n-1)), 34);
```



```
                    Wnext( (Xind(2*n)), 21:24) = W( sel(Xind(2*n)), 21:24);
                    Wnext( (Xind(2*n)), 34) = W( sel(Xind(2*n)), 34);
                end

            else% gene == 7;    %out3 25:28; 35
                if Xgene(gene, 1) < pcross; %cross
                    Wnext( (Xind(2*n-1)), 25:28) = W( sel(Xind(2*n)), 25:28);
                    Wnext( (Xind(2*n-1)), 35) = W( sel(Xind(2*n)), 35);
                    Wnext( (Xind(2*n)), 25:28) = W( sel(Xind(2*n-1)), 25:28);
                    Wnext( (Xind(2*n)), 35) = W( sel(Xind(2*n-1)), 35);
                else %copy
                    Wnext( (Xind(2*n-1)), 25:28) = W( sel(Xind(2*n-1)), 25:28);
                    Wnext( (Xind(2*n-1)), 35) = W( sel(Xind(2*n-1)), 35);
                    Wnext( (Xind(2*n)), 25:28) = W( sel(Xind(2*n)), 25:28);
                    Wnext( (Xind(2*n)), 35) = W( sel(Xind(2*n)), 35);
                end

            end
        end
    end

    pmut = 0.001;
    %mutation--------------------------------------
    for n = 41:100; %mutated top 60 genes
        if rand() < pmut; %MUTATE
            pos1 = randperm(35);
            pos = pos1(1);
            if pos < 35 %mutate some bits
                Wnext(n, 1:pos) = randn(1, 1:pos)+W(sel(n-40), 1:pos);
                Wnext(n, (pos+1):35);
            else %mutate all bits
                Wnext(n, 1:35) = randn(1, 1:35)+W(sel(n-40), 1:35);
            end
        else
            Wnext(n, 1:35) = W(sel(n-40), 1:35);
        end
    end

    %update next weights
    W = Wnext;
end %end training
trainingt = toc;

Wbest = W(sel(1), :);
scorrect = 0; vccorrect = 0; vergcorrect = 0; nerror = 0;
serror = 0; vcerror = 0; vergerror = 0; nerror = 0;

disp(' ')
fprintf(1,' Iris-setosa is represented by output:      %.0f \n',setosa);
fprintf(1,' Iris-versicolor is represented by output:  %.0f \n',versicolor);
fprintf(1,' Iris-verginica is represented by output:   %.0f \n',verginica);
```

```
disp(' ')
disp(' Hit any key to test the network using the test data set.')
disp(' ')
pause
% Testing-----------------------------------------
%use test_p   & test_t
tic;
for tn = 1:testingsize
    %weights (to, from)
    %   [row 1 = from first layer
    %    column 1 = to first layer]
    % to hidden[1,4] from input[1,3]
    %   [           W(1)                    W(2)                    W(3) ↙
W(4);          W(29)    %hidden1
    %              W1'1                    W2'1                    W3'1 ↙
W4'1;         bias
    h1=test_p(1,tn)*Wbest(1,1)+ test_p(2,tn)*Wbest(1,2)+ test_p(2,tn)*Wbest(1,3)+ test_p ↙
(4,tn)*Wbest(1,4)+Wbest(1,29);
    z1= 1/(1+exp(-h1));

       %                W(5)                W(6)                    W(7) ↙
W(8);          W(30)    %hidden2
       %                W1'2                W2'2                    W3'2 ↙
W4'2;          bias
    h2=test_p(1,tn)*Wbest(1,5)+ test_p(2,tn)*Wbest(1,6)+ test_p(2,tn)*Wbest(1,7)+ test_p ↙
(4,tn)*Wbest(1,8)+Wbest(1,3);
    z2= 1/(1+exp(-h2));

    %                   W(9)                    W(10)                       W(11) ↙
W(12)       ;   W(31)   %hidden3
    %    W1'3 W2'3 W3'3 W4'3;   bias
    h3=test_p(1,tn)*Wbest(1,9)+ test_p(2,tn)*Wbest(1,10)+ test_p(2,tn)*Wbest(1,11)+ ↙
test_p(4,tn)*Wbest(1,12)+Wbest(1,31);
    z3= 1/(1+exp(-h3));

    %                   W(13)                   W(14)                       W(15) ↙
W(16)          W(32) %hidden4
    %    W1'4 W2'4 W3'4 W4'4;
    h4=test_p(1,tn)*Wbest(1,13)+ test_p(2,tn)*Wbest(1,14)+ test_p(2,tn)*Wbest(1,15)+ ↙
test_p(4,tn)*Wbest(1,16)+Wbest(1,32);
    z4= 1/(1+exp(-h4));

    %to output (layer 3) from hidden (layer 2)
    %              [W15                    W25             W35             W45 ↙
T5; out1
    %               W(17)                W(18)           W(19)           W(20)        W ↙
(33);
    out1 = z1*Wbest(1, 17) + z2*Wbest(1, 18)+ z3*Wbest(1, 19)+ z4*Wbest(1, 20)+ Wbest(1, ↙
33);
```



```
    %            W16             W26             W36             W46 ↙
T6;
    %           W(21)            W(22)           W(23)          W(24)            W ↙
(34); out2
    out2 = z1*Wbest(1, 21) + z2*Wbest(1, 22)+ z3*Wbest(1, 23)+ z4*Wbest(1, 24)+ Wbest(1, ↙
34);

    %             W17             W27             W37             W47 ↙
T7];
    %            W(25)            W(26)           W(27)           W(28)          W ↙
(35)];
    out3 = z1*Wbest(1, 25) + z2*Wbest(1, 26)+ z3*Wbest(1, 27)+ z4*Wbest(1, 28)+ Wbest(1, ↙
35);

    %get simulated outputs Y from defined NN net with defined weights W,
    %and input P
    if max([out1 out2 out3]) == out1;
        Yf = [1; 0; 0];
    elseif max([out1, out2, out3]) == out2;
        Yf = [0; 1; 0];
    else% max([out1, out2, out3]) == out3;
        Yf = [0; 0; 1];
    end
    Yf

%     if Yf(1) == test_t(1,tn) && Yf(2) == test_t(2,tn) && Yf(3) == test_t(3,tn) && tn <= ↙
endsetosa
%         scorrect = scorrect + 1;
%     elseif Yf(1) == test_t(1,tn) && Yf(2) == test_t(2,tn) && Yf(3) == test_t(3,tn) && ↙
tn <= endvcolor
%         vccorrect = vccorrect + 1;
%     elseif Yf(1) == test_t(1,tn) && Yf(2) == test_t(2,tn) && Yf(3) == test_t(3,tn) && ↙
tn <= endverg
%         vergcorrect = vergcorrect +1;
%     else nerror = nerror + 1;
%     end
    if tn > endvcolor && (Yf(1) ~= test_t(1,tn) || Yf(2) ~= test_t(2,tn) || Yf(3) ~= ↙
test_t(3,tn))
        vergerror = vergerror +1;
    elseif tn > endsetosa && (Yf(1) ~= test_t(1,tn) || Yf(2) ~= test_t(2,tn) || Yf(3) ~= ↙
test_t(3,tn))
        vcerror = vcerror + 1;
    elseif tn <= endsetosa && (Yf(1) ~= test_t(1,tn) || Yf(2) ~= test_t(2,tn) || Yf(3) ~= ↙
test_t(3,tn))
        serror = serror + 1;
%     else nerror = nerror + 1;
    end
    Y = [out1; out2; out3];

    %Error = expected - simulated = T - Y
    E(tn,:) = test_t(:,tn) - Y;
```



```
    msE(tn,:) = mse(E(tn,:)); %mean squared error
end
testingt = toc;

serror = serror/nsetosa*100
vcerror = vcerror/nvcolor*100;
vergerror = vergerror/nverg*100;
nerror = nerror/testingsize*100;
terror = nerror + vergerror + vcerror + serror;

% serror = abs(scorrect-nsetosa)/nsetosa*100;
% vcerror = abs(vccorrect-nvcolor)/nvcolor*100;
% vergerror = abs(vergcorrect-nverg)/nverg*100;
% nerror = abs(nerror-testingsize)/testingsize*100;
% terror = nerror + vergerror + vcerror + serror;

fprintf(1,' Training time:                    %.2f \n', trainingt);
fprintf(1,' Testing time:                     %.2f \n', testingt);
fprintf(1,' \n')
fprintf(1,' Iris-setosa recognition error:    %.2f \n',serror);
fprintf(1,' Iris-versicolor recognition error:  %.2f \n',vcerror);
fprintf(1,' Iris-verginica recognition error:   %.2f \n',vergerror);
fprintf(1,' missclassification error:   %.2f \n',nerror);
fprintf(1,' \n')
fprintf(1,' Total Iris plant recognition error: %.2f \n',terror);
fprintf(1,' \n')
```



```
%Edris Evolutionary NN
%inputs = 4
%weights = 20

%hidden = 5
%bias = 5
%weights = 15

%out = 3
%bias = 3

%total weights = 43
% function [msE] = myNN5(W)

% W=rand(100, 27);
% W=rand(100, 35);
echo off;
rng('default');
W=randn(100, 43);
W1 = W;

%input training data
[iris_data] = Iris_data;
iris_data = (iris_data(:,[1:4]))';

for n=1:4; %scaling
    iris_inputs(n,:)=(iris_data(n,:)-min(iris_data(n,:)))/ ...
        (max(iris_data(n,:)-min(iris_data(n,:))));
end

%target training data
iris_target1 = [1 0 0]'; setosa=find(iris_target1);
iris_target2 = [0 1 0]'; versicolor=find(iris_target2);
iris_target3 = [0 0 1]'; verginica=find(iris_target3);

for n=1:(50-1)
   iris_target1=[iris_target1 iris_target1(:,1)];
   iris_target2=[iris_target2 iris_target2(:,1)];
   iris_target3=[iris_target3 iris_target3(:,1)];
end

iris_targets = [iris_target1 iris_target2 iris_target3];

p=[]; t=[]; test_p=[]; test_t=[]; setosaind=[]; versicolorind=[]; verginicaind=[]; amsE= ↙
[];

testc = 0;
for n=1:150
   if rand(1)>1/3   %trainging data
      p=[p iris_inputs(:,n)];
      t=[t iris_targets(:,n)];
```

```
   else            %testing data
      testc = testc+1; %increment test index
      test_p=[test_p iris_inputs(:,n)];
      test_t=[test_t iris_targets(:,n)];
      if n<51
          setosaind=[setosaind n];
          endsetosa = testc; %index of test where setosaind ends
      elseif n<101
          versicolorind=[versicolorind n];
          endvcolor = testc; %index of test where versicolorind ends
      else
          verginicaind=[verginicaind n];
          endverg = testc;  %index of test where verginicaind ends
      end
   end
end
[a nsetosa] = size(setosaind)
[a nvcolor] = size(versicolorind)
[a nverg] = size(verginicaind)
% %P is input data for NN
% P = iris_data;
%
% %T is training output for NN
% T = vertcat(iris_target1', iris_target2', iris_target3')';

[m n]=size(test_p);

trainingsize = 150 - n;
testingsize = n;

disp(' ')
fprintf(1,' The training data set contains %.0f elements.\n',(150-n));
fprintf(1,' The test data set contains %.0f elements.\n',n);
disp(' ')

chromo = 1;
train = 1;
tic;
for train = 1:trainingsize;
    for chromo = 1:100;
        %weights (to, from)
        %   [row 1 = from first layer
        %    column 1 = to first layer]
        % to hidden[1,5] from input[1,3]
        %   [W(1) W(2) W(3) W(4); W(36)bias   %hidden1
        %    W1'1 W2'1 W3'1 W4'1;
        h1=p(1,train)*W(chromo,1)+ p(2,train)*W(chromo,2)+ p(2,train)*W(chromo,3)+ p(4, ↙
train)*W(chromo,4)+W(chromo,36);
        z1= 1/(1+exp(-h1));

        %    W(5) W(6) W(7) W(8);    %hidden2
```



```
        %    W1'2 W2'2 W3'2 W4'2;
        h2=p(1,train)*W(chromo,5)+ p(2,train)*W(chromo,6)+ p(2,train)*W(chromo,7)+ p(4, ↙
train)*W(chromo,8)+W(chromo,37);
        z2= 1/(1+exp(-h2));

        %    W(9) W(10) W(11) W(12);    %hidden3
        %    W1'3 W2'3 W3'3 W4'3;
        h3=p(1,train)*W(chromo,9)+ p(2,train)*W(chromo,10)+ p(2,train)*W(chromo,11)+ p(4, ↙
train)*W(chromo,12)+W(chromo,38);
        z3= 1/(1+exp(-h3));

        %    W(13) W(14) W(15) W(16)    %hidden4
        %    W1'4 W2'4 W3'4 W4'4;
        h4=p(1,train)*W(chromo,13)+ p(2,train)*W(chromo,14)+ p(2,train)*W(chromo,15)+ p ↙
(4,train)*W(chromo,16)+W(chromo,39);
        z4= 1/(1+exp(-h4));

        %    W(17) W(18) W(19) W(20)];    %hidden5
        %               W1'5                    W2'5                    W3'5 ↙
W4'5]
        h5=p(1,train)*W(chromo,17)+ p(2,train)*W(chromo,18)+ p(2,train)*W(chromo,19)+ p ↙
(4,train)*W(chromo,20)+W(chromo,40);
        z5= 1/(1+exp(-h5));

        %to output (layer 3) from hidden (layer 2)
        %   [W16                    W26                 W36              W46 ↙
W56                T6; out1
        %         W(21)                W(22)               W(23)           W(24) ↙
W(25)               W(41);
        out1 = z1*W(chromo, 21) + z2*W(chromo, 22)+ z3*W(chromo, 23)+ z4*W(chromo, 24)+ ↙
z5*W(chromo, 35)+ W(chromo, 41);

        %                   W17               W27               W37             W47 ↙
W57          T7;
        %                 W(26)               W(27)             W(28)          W(29) ↙
W(30)       W(42); out2
        out2 = z1*W(chromo, 26) + z2*W(chromo, 27)+ z3*W(chromo, 28)+ z4*W(chromo, 29)+ ↙
z5*W(chromo, 30)+ W(chromo, 42);

        %    W18   W28   W38   W48   W58   T8];
        %                W(31)              W(32)             W(33)             W(34) ↙
W(35)       W(43)];
        out3 = z1*W(chromo, 31) + z2*W(chromo, 32)+ z3*W(chromo, 33)+ z4*W(chromo, 34)+ ↙
z5*W(chromo, 35)+ W(chromo, 43);

        %get simulated outputs Y from defined NN net with defined weights W,
        %and input P
        Y = [out1; out2; out3];

        %Error = expected - simulated = T - Y
        E(chromo,:) = t(:,train) - Y;
```



```
        msE(chromo,:) = mse(E(chromo,:));  %mean squared error

    end %end population

    amsE = [amsE, train];
    if train > 1
        if amsE(train) < 0.07
            break %stop training
        end
    end
    Wnext = zeros(100, 43); %next population
    Esearch = msE;    %used for selection to find smallest error

    %selection order of E index (least E) chromosomes----------------
    %this array order least mse to worst mse indcies
    %which coorespond to W indecies
    for i = 1:100;
        [val, ind] = min(Esearch);
        sel(i,1) = ind;
        Esearch(ind, 1) = 100;
    end
%     sel; %array of indecies of E order min - max

    %crossover-------------------------------
    pcross = 0.2;
    Xind = randperm(40); %select order of top 40 indcies to be crossed
    Xgene = rand(8,1); %probablilty or crossover per gene
    for n = 1:20;
        for gene = 1:8;
            if gene == 1;    %hidden 1 1:4; 36
                if Xgene(gene, 1) < pcross; %cross
                    Wnext( (Xind(2*n-1)), 1:4)  = W( sel(Xind(2*n)), 1:4);
                    Wnext( (Xind(2*n-1)), 36)   = W( sel(Xind(2*n)), 36);
                    Wnext( (Xind(2*n)), 1:4)    = W( sel(Xind(2*n-1)), 1:4);
                    Wnext( (Xind(2*n)), 36)     = W( sel(Xind(2*n-1)), 36);
                else %copy
                    Wnext( (Xind(2*n-1)), 1:4) = W( sel(Xind(2*n-1)), 1:4);
                    Wnext( (Xind(2*n-1)), 36) = W( sel(Xind(2*n-1)), 36);
                    Wnext( (Xind(2*n)), 1:4) = W( sel(Xind(2*n)), 1:4);
                    Wnext( (Xind(2*n)), 36) = W( sel(Xind(2*n)), 36);

                end

            elseif gene == 2;    %hidden 2 5:8; 37
                if Xgene(gene, 1) < pcross; %cross
                    Wnext( (Xind(2*n-1)), 5:8) = W( sel(Xind(2*n)), 5:8);
                    Wnext( (Xind(2*n-1)), 37) = W( sel(Xind(2*n)), 37);
                    Wnext( (Xind(2*n)), 5:8) = W( sel(Xind(2*n-1)), 5:8);
                    Wnext( (Xind(2*n)), 37) = W( sel(Xind(2*n-1)), 37);
                else %copy
                    Wnext( (Xind(2*n-1)), 5:8) = W( sel(Xind(2*n-1)), 5:8);
```



```
                    Wnext( (Xind(2*n-1)), 37) = W( sel(Xind(2*n-1)), 37);
                    Wnext( (Xind(2*n)), 5:8) = W( sel(Xind(2*n)), 5:8);
                    Wnext( (Xind(2*n)), 37) = W( sel(Xind(2*n)), 37);
                end

            elseif gene == 3;    %hidden 3 9:12; 38;
                if Xgene(gene, 1) < pcross; %cross
                    Wnext( (Xind(2*n-1)), 9:12) = W( sel(Xind(2*n)), 9:12);
                    Wnext( (Xind(2*n-1)), 38) = W( sel(Xind(2*n)), 38);
                    Wnext( (Xind(2*n)), 9:12) = W( sel(Xind(2*n-1)), 9:12);
                    Wnext( (Xind(2*n)), 38) = W( sel(Xind(2*n-1)), 38);
                else %copy
                    Wnext( (Xind(2*n-1)), 9:12) = W( sel(Xind(2*n-1)), 9:12);
                    Wnext( (Xind(2*n-1)), 38) = W( sel(Xind(2*n-1)), 38);
                    Wnext( (Xind(2*n)), 9:12) = W( sel(Xind(2*n)), 9:12);
                    Wnext( (Xind(2*n)), 38) = W( sel(Xind(2*n)), 38);
                end

            elseif gene == 4;    %hidden 4 13:16; 39
                if Xgene(gene, 1) < pcross; %cross
                    Wnext( (Xind(2*n-1)), 13:16) = W( sel(Xind(2*n)), 13:16);
                    Wnext( (Xind(2*n-1)), 39) = W( sel(Xind(2*n)), 39);
                    Wnext( (Xind(2*n)), 13:16) = W( sel(Xind(2*n-1)), 13:16);
                    Wnext( (Xind(2*n)), 39) = W( sel(Xind(2*n-1)), 39);
                else %copy
                    Wnext( (Xind(2*n-1)), 13:16) = W( sel(Xind(2*n-1)), 13:16);
                    Wnext( (Xind(2*n-1)), 39) = W( sel(Xind(2*n-1)), 39);
                    Wnext( (Xind(2*n)), 13:16) = W( sel(Xind(2*n)), 13:16);
                    Wnext( (Xind(2*n)), 39) = W( sel(Xind(2*n)), 39);
                end

            elseif gene == 5;    %hidden 5 17:20; 40
                if Xgene(gene, 1) < pcross; %cross
                    Wnext( (Xind(2*n-1)), 17:20) = W( sel(Xind(2*n)), 17:20);
                    Wnext( (Xind(2*n-1)), 40) = W( sel(Xind(2*n)), 40);
                    Wnext( (Xind(2*n)), 17:20) = W( sel(Xind(2*n-1)), 17:20);
                    Wnext( (Xind(2*n)), 40) = W( sel(Xind(2*n-1)), 40);
                else %copy
                    Wnext( (Xind(2*n-1)), 17:20) = W( sel(Xind(2*n-1)), 17:20);
                    Wnext( (Xind(2*n-1)), 40) = W( sel(Xind(2*n-1)), 40);
                    Wnext( (Xind(2*n)), 17:20) = W( sel(Xind(2*n)), 17:20);
                    Wnext( (Xind(2*n)), 40) = W( sel(Xind(2*n)), 40);
                end

            elseif gene == 6;    %out1 21:25; 41
                if Xgene(gene, 1) < pcross; %cross
                    Wnext( (Xind(2*n-1)), 21:25) = W( sel(Xind(2*n)), 21:25);
                    Wnext( (Xind(2*n-1)), 41) = W( sel(Xind(2*n)), 41);
                    Wnext( (Xind(2*n)), 21:25) = W( sel(Xind(2*n-1)), 21:25);
                    Wnext( (Xind(2*n)), 41) = W( sel(Xind(2*n-1)), 41);
                else %copy
```

```
                Wnext( (Xind(2*n-1)), 21:25) = W( sel(Xind(2*n-1)), 21:25);
                Wnext( (Xind(2*n-1)), 41) = W( sel(Xind(2*n-1)), 41);
                Wnext( (Xind(2*n)), 21:25) = W( sel(Xind(2*n)), 21:25);
                Wnext( (Xind(2*n)), 41) = W( sel(Xind(2*n)), 41);
            end

        elseif gene == 7;    %out2 26:30; 42
            if Xgene(gene, 1) < pcross; %cross
                Wnext( (Xind(2*n-1)), 26:30) = W( sel(Xind(2*n)), 26:30);
                Wnext( (Xind(2*n-1)), 42) = W( sel(Xind(2*n)), 42);
                Wnext( (Xind(2*n)), 26:30) = W( sel(Xind(2*n-1)), 26:30);
                Wnext( (Xind(2*n)), 42) = W( sel(Xind(2*n-1)), 42);
            else %copy
                Wnext( (Xind(2*n-1)), 26:30) = W( sel(Xind(2*n-1)), 26:30);
                Wnext( (Xind(2*n-1)), 42) = W( sel(Xind(2*n-1)), 42);
                Wnext( (Xind(2*n)), 26:30) = W( sel(Xind(2*n)), 26:30);
                Wnext( (Xind(2*n)), 42) = W( sel(Xind(2*n)), 42);
            end

        else    % gene == 8  out3 31:35; 43
            if Xgene(gene, 1) < pcross; %cross
                Wnext( (Xind(2*n-1)), 31:35) = W( sel(Xind(2*n)), 31:35);
                Wnext( (Xind(2*n-1)), 43) = W( sel(Xind(2*n)), 43);
                Wnext( (Xind(2*n)), 31:35) = W( sel(Xind(2*n-1)), 31:35);
                Wnext( (Xind(2*n)), 43) = W( sel(Xind(2*n-1)), 43);
            else %copy
                Wnext( (Xind(2*n-1)), 31:35) = W( sel(Xind(2*n-1)), 31:35);
                Wnext( (Xind(2*n-1)), 43) = W( sel(Xind(2*n-1)), 43);
                Wnext( (Xind(2*n)), 31:35) = W( sel(Xind(2*n)), 31:35);
                Wnext( (Xind(2*n)), 43) = W( sel(Xind(2*n)), 43);
            end
        end
    end
end

pmut = 0.001;
%mutation--------------------------------------
for n = 41:100; %mutated top 60 genes
    if rand() < pmut; %MUTATE
        pos1 = randperm(43);
        pos = pos1(1);
        if pos < 43 %mutate some bits
            Wnext(n, 1:pos) = randn(1, 1:pos)+W(sel(n-40), 1:pos);
            Wnext(n, (pos+1):43);
        else %mutate all bits
            Wnext(n, 1:43) = randn(1, 1:43)+W(sel(n-40), 1:43);
        end
    else
        Wnext(n, 1:43) = W(sel(n-40), 1:43);
    end
end
```



```
    %update next weights
    W = Wnext;
end %end training
trainingt = toc;

Wbest = W(sel(1), :);
scorrect = 0; vccorrect = 0; vergcorrect = 0; nerror = 0;
serror = 0; vcerror = 0; vergerror = 0; nerror = 0;

disp(' ')
fprintf(1,' Iris-setosa is represented by output:      %.0f \n',setosa);
fprintf(1,' Iris-versicolor is represented by output:  %.0f \n',versicolor);
fprintf(1,' Iris-verginica is represented by output:   %.0f \n',verginica);

disp(' ')
disp(' Hit any key to test the network using the test data set.')
disp(' ')
pause
% Testing-----------------------------------------------
%use test_p   & test_t
tic;
for tn = 1:testingsize
    %weights (to, from)
    %   [row 1 = from first layer
    %    column 1 = to first layer]
    % to hidden[1,4] from input[1,3]
    %   [W(1) W(2) W(3) W(4); W(36)bias   %hidden1
    %    W1'1 W2'1 W3'1 W4'1;
    h1=test_p(1,tn)*Wbest(1,1)+ test_p(2,tn)*Wbest(1,2)+ test_p(2,tn)*Wbest(1,3)+ test_p
(4,tn)*Wbest(1,4)+Wbest(1,36);
    z1= 1/(1+exp(-h1));

    %    W(5) W(6) W(7) W(8);    %hidden2
    %    W1'2 W2'2 W3'2 W4'2;
    h2=test_p(1,tn)*Wbest(1,5)+ test_p(2,tn)*Wbest(1,6)+ test_p(2,tn)*Wbest(1,7)+ test_p
(4,tn)*Wbest(1,8)+Wbest(1,37);
    z2= 1/(1+exp(-h2));

    %    W(9) W(10) W(11) W(12);    %hidden3
    %    W1'3 W2'3 W3'3 W4'3;
    h3=test_p(1,tn)*Wbest(1,9)+ test_p(2,tn)*Wbest(1,10)+ test_p(2,tn)*Wbest(1,11)+
test_p(4,tn)*Wbest(1,12)+Wbest(1,38);
    z3= 1/(1+exp(-h3));

    %    W(13) W(14) W(15) W(16)    %hidden4
    %    W1'4 W2'4 W3'4 W4'4;
    h4=test_p(1,tn)*Wbest(1,13)+ test_p(2,tn)*Wbest(1,14)+ test_p(2,tn)*Wbest(1,15)+
test_p(4,tn)*Wbest(1,16)+Wbest(1,39);
    z4= 1/(1+exp(-h4));
```



```
    %    W(17) W(18) W(19) W(20)];    %hidden5
    %               W1'5                    W2'5                    W3'5
W4'5]
    h5=test_p(1,tn)*Wbest(1,17)+ test_p(2,tn)*Wbest(1,18)+ test_p(2,tn)*Wbest(1,19)+
test_p(4,tn)*Wbest(1,20)+Wbest(1,40);
    z5= 1/(1+exp(-h5));

    %to output (layer 3) from hidden (layer 2)
    %   [W16                    W26                 W36              W46            W56
T6; out1
    %         W(21)                 W(22)               W(23)           W(24)           W
(25)              W(41);
    out1 = z1*Wbest(1, 21) + z2*Wbest(1, 22)+ z3*Wbest(1, 23)+ z4*Wbest(1, 24)+ z5*Wbest
(1, 35)+ Wbest(1, 41);

    %                   W17               W27               W37             W47
W57          T7;
    %               W(26)               W(27)              W(28)          W(29)
W(30)       W(42); out2
    out2 = z1*Wbest(1, 26) + z2*Wbest(1, 27)+ z3*Wbest(1, 28)+ z4*Wbest(1, 29)+ z5*Wbest
(1, 30)+ Wbest(1, 42);

    %    W18   W28   W38   W48   W58   T8];
    %                W(31)              W(32)             W(33)             W(34)
W(35)        W(43)];
    out3 = z1*Wbest(1, 31) + z2*Wbest(1, 32)+ z3*Wbest(1, 33)+ z4*Wbest(1, 34)+ z5*Wbest
(1, 35)+ Wbest(1, 43);

    %get simulated outputs Y from defined NN net with defined weights W,
    %and input P
    if max([out1 out2 out3]) == out1;
        Yf = [1; 0; 0];
    elseif max([out1, out2, out3]) == out2;
        Yf = [0; 1; 0];
    else% max([out1, out2, out3]) == out3;
        Yf = [0; 0; 1];
    end
    Yf

%     if Yf(1) == test_t(1,tn) && Yf(2) == test_t(2,tn) && Yf(3) == test_t(3,tn) && tn <=
endsetosa
%         scorrect = scorrect + 1;
%     elseif Yf(1) == test_t(1,tn) && Yf(2) == test_t(2,tn) && Yf(3) == test_t(3,tn) &&
tn <= endvcolor
%         vccorrect = vccorrect + 1;
%     elseif Yf(1) == test_t(1,tn) && Yf(2) == test_t(2,tn) && Yf(3) == test_t(3,tn) &&
tn <= endverg
%         vergcorrect = vergcorrect +1;
%     else nerror = nerror + 1;
%     end
    if tn > endvcolor && (Yf(1) ~= test_t(1,tn) || Yf(2) ~= test_t(2,tn) || Yf(3) ~=
```



```
test_t(3,tn))
        vergerror = vergerror +1;
    elseif tn > endsetosa && (Yf(1) ~= test_t(1,tn) || Yf(2) ~= test_t(2,tn) || Yf(3) ~=
test_t(3,tn))
        vcerror = vcerror + 1;
    elseif tn <= endsetosa && (Yf(1) ~= test_t(1,tn) || Yf(2) ~= test_t(2,tn) || Yf(3) ~=
test_t(3,tn))
        serror = serror + 1;
%     else nerror = nerror + 1;
    end
    Y = [out1; out2; out3];

    %Error = expected - simulated = T - Y
    E(tn,:) = test_t(:,tn) - Y;
    msE(tn,:) = mse(E(tn,:)); %mean squared error
end
testingt = toc;

serror = serror/nsetosa*100
vcerror = vcerror/nvcolor*100;
vergerror = vergerror/nverg*100;
nerror = nerror/testingsize*100;
terror = nerror + vergerror + vcerror + serror;

% serror = abs(scorrect-nsetosa)/nsetosa*100;
% vcerror = abs(vccorrect-nvcolor)/nvcolor*100;
% vergerror = abs(vergcorrect-nverg)/nverg*100;
% nerror = abs(nerror-testingsize)/testingsize*100;
% terror = nerror + vergerror + vcerror + serror;

fprintf(1,' Training time:                    %.2f \n', trainingt);
fprintf(1,' Testing time:                     %.2f \n', testingt);
fprintf(1,' \n')
fprintf(1,' Iris-setosa recognition error:    %.2f \n',serror);
fprintf(1,' Iris-versicolor recognition error:  %.2f \n',vcerror);
fprintf(1,' Iris-verginica recognition error:   %.2f \n',vergerror);
fprintf(1,' missclassification error:   %.2f \n',nerror);
fprintf(1,' \n')
fprintf(1,' Total Iris plant recognition error: %.2f \n',terror);
fprintf(1,' \n')
```